UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/657,463	09/08/2003	**Makarand Gadre**	MS1-1596US	9824

22801 7590 08/15/2008
LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

EXAMINER	
WEI, ZHENG	
ART UNIT	**PAPER NUMBER**
2192	
MAIL DATE	**DELIVERY MODE**
08/15/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/657,463	GADRE, MAKARAND	
	Examiner	Art Unit	
	ZHENG WEI	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *28 May 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-9,11-40 and 42-54* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-9, 11-40 and 42-54* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *05/28/2008*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

Detailed Action

Remarks

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 05/28/2008 has been entered.
2. This office action is in response to the amendment filed on 05/11/2008.
3. Claim 41 has been cancelled
4. Claims 1, 2, 4, 8, 13, 18, 25, 31-36, 40, 46, 48, 50 and 51 have been amended.
5. Claims 1-9, 11-40, and 42-54 remain pending and have been examined.

Information Disclosure Statement

6. The information disclosure statements filed on 05/28/2008 has been placed in the application file and the information referred to therein has already been considered.

Response to Arguments

7. The Applicants amendment to independent claims 1, 13, 18, 25 31, 36, 40 and 51 change the scope of claims. Therefore a new ground of rejection is applied.

Double Patenting

8. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

9. Claims 1, 13 and 18 are provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 1, 12 and 23 of copending Application No. 10/657,468. Although the conflicting claims are not identical, they are not patentably distinct from each other. As can be seen from the table below, instant claims and the claims of copending application are directed to the same subject matter of the invention. For example,

Instant Application 10/657,463	**Copending Application** 10/657,468
Claim 1. **A method of generating common intermediate language code comprising**:	Claim 1. **A method of generating common intermediate language code comprising**:
receiving a portion of JAVA.TM. …	writing first object oriented language source code that comprises a definition of a generic class usable in a framework,
Generating, through a first compiler different from a formal compiler complying with the formal JAVA.TM language specification **language-neutral intermediate language code** representing the portion of JAVA language source code referencing the one or more generic classes	compiling the instance of the generic class into **common intermediate language code** executable by a runtime engine.

Claim 13. A method of compiling comprising: **receiving a portion of JAVA.TM. language** … **creating an intermediate language code** …	Claim 12. A method of using a generic class comprising: **adapting existing object oriented language source code,** the existing object oriented language source code being associated with a framework developed for use with a predetermined object oriented programming language that compiles source code into non-language-neutral bytecodes, to include a declaration of a first generic class provided by a software package having a class definition of the first generic class; and compiling the adapted existing object oriented language source code with the class definition to **generate common intermediate language code**
Claim 18. A computer-readable medium having stored thereon computer-executable instructions for performing a method of compiling comprising: **receiving a portion of JAVA.TM. language** … **creating a parse tree having a generic class identifier associated with the generic class** and type identifier associated with the specified type; and generating one or more **intermediate language instructions** representing the JAVA.TM. M language instruction based	Claim 23. A computer-readable medium having stored thereon microprocessor-executable instructions for performing a method comprising: **receiving input representing a generic class definition in a JAVA.TM. language**, the first object oriented language being associated with a framework developed for use with a predetermined object oriented programming language that compiles source code into non-language-neutral bytecodes; receiving a second, different object oriented language defined by the first object oriented language source code; and **compiling the source code** with **an**

on the parse tree based on the portion of JAVA language source code and the referenced generic class provided in the programming language other than JAVA language.	**instance of the generic class** into common intermediate language code executable by a runtime engine.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 1-9, 13-30, 36-40, 42-45 and 50-54 are rejected under 35 U.S.C. 103(a) as being unpatentable over by Hostetter (Hostetter et al., US 2005/0060695 A1) in view of Allen (Eric Allen, "Diagnosing Java code: Java generics without the pain, part 1-part2) and in further view of Getov (Getov et al., "High-performance parallel programming in Java exploiting native libraries")

Claim 1:

Hostetter discloses a method of generating common intermediate language code for user in a framework, the method comprising:

- receiving a portion of JAVA.TM. language source code referencing, through a generic class syntax, one or more generic classes, unspecified in a formal JAVA.TM language specification (see for example, Fig.5, step 2020 "Read Instruction From Program Code" and related text; also see p.2, [0016]-[0017], "Java programming language", "template generated class"), wherein:
- Generating, , language-neutral intermediate language code representing the portion of JAVA.TM language source code referencing the one or more generic class (see for example, Fig.5, step 2045, 2075, "Compile" and related text)

But does not explicitly disclose receiving one or more generic classes through a generic class syntax unspecified in a formal JAVA.TM language specification. However, Allen in the same analogous art of Java generics, discloses

- each of the one or more generic classes refers to a first class configured to operate uniformly on values of different types associated with the first class and defined by a plurality of second classes (see for example, p.2, example of defining a class Hashtable as a generic class; also see Listing 4 and related text);
- the generic class syntax is not specified in the formal JAV.TM language specification and identifies one of the plurality of second class by surrounding the one of the plurality of second classes with angular brackets following the first class (see for example, p.2, section "Generic

types to the rescue", "The syntax for defining such generic classes in Tiger is just like that for ordinary class definition..."); and

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to Allen's method to implement Hostetter's parameterized class template. One would have been motivated to do so to eliminate casts and errors as suggested by Allen (see for example, p.1-2, section Casts and error and section "Generic types to the rescue" "A natural way to eliminate casts like the one above is to argument the Java type system with what are known as generic types")

Hostetter and Allen do not explicitly disclose generating language-neutral intermediate language code representing the portion of source codereferenceing the one or more generic classes through a first compiler different from a formal compiler complying with the formal JAVA.TM language specification. However, Getov in the same analogous art of programming in Java, discloses Java code can use high performance classes (libraries) written in other language (see for example, p.1, SUMMARY, "in this paper we discuss how new Java programs can capitalize on high-performance libraries for other language" and related descriptions). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to compile the other classes/libraries in a programming language (Fortran, C) other than Java language by using a compiler different from a formal compiler complying with the formal JAVA language specification. One would have been motivated to do so to

enable high-performance computation in Java by reference existing native libraries in other programming language.

Claim 2:

Hostetter further discloses a method as recited in claim 1, comprising parsing the portion of the source code into a parse tree representing the portion of the JAVA.TM language source code before compiling the portion with the first class (see for example, Fig.5, step 2045, 2075, "Compile" and related text; also see p.2, [0024], [0026], "The runtime compiler is invoked...").

Claim 3:

Hostetter also discloses a method as recited in claim 2 further comprising nesting a constructed class of the first class in the parse tree (see for example, p.2, [0022] "(2) creating a representation of the template-generated class...").

Claim 4:

Hostetter further discloses a method as recited in claim 1 further comprising:

- generating a parse tree having a token referencing the first class and a token referencing the specified one of the plurality of associated classes (see for example, Fig.5, step 2045, 2075, "Compile"; also see p.2 [0024]); and
- semantically analyzing the parse tree to determine validity of semantics of the first class (see for example, Fig.5, step 2085, "Compiler Error"; also see,

p.6, [0090], "the compiler notifies the software programmer or end user of the compilation error…").

Claim 5:

Hostetter also discloses a method as recited in claim 4 wherein the semantically analyzing comprises determining whether operations applied to the first class are valid (see for example, Fig.5, step 2085, "Compiler Error"; also see, p.6, [0090], "the compiler notifies the software programmer or end user of the compilation error…").

Claim 6:

Hostetter further discloses a method as recited in claim 1 further comprising generating metadata descriptive of the first class (see for example, Fig.2, step 1020, "Create Method Bindings for Class Representation" and related text; also see p.4, [0055], "A method binding is an object that stores information about a class method").

Claim 7:

Hostetter also discloses a method as recited in claim 6 further comprising storing the metadata with the language-neutral intermediate language code, whereby the language-neutral intermediate language code is used by an application program (see for example, Fig.2, step 1010, "Create object Representation of Template-

Generated Class" and related text; also see, p.4, [0053], "template representation").

Claim 8:

Hostetter further discloses a method as recited in claim 1 further comprising creating a compiled project (see for example, Fig.2, step 1020, 1030, "Compile"; also see p.4, [0054], "compilation") including the language-neutral intermediate language code (see for example, Fig.2, step 1010, "Create object Representation of Template-Generated Class" and related text; also see, p.4, [0053], "template representation") and metadata descriptive of the first class and the specified one of the plurality of second classes (see for example, Fig.2, step 1020, "Create Method Bindings for Class Representation" and related text; also see p.4, [0055], "A method binding is an object that stores information about a class method").

Claim 9:

Hostetter further discloses a method as recited in claim 1 further comprising executing the language-neutral intermediate language code with a runtime engine (see for example, p.4, [0054], "executable code").

Claim 13:

Incorporating with rejection to Claim 1, Hostetter further discloses a method of compiling in a framework, the method comprising:

- parsing the declaration into a token corresponding to the generic class (see for example, p.2, [0024] "requires compilation"); and
- creating an intermediate language code block corresponding to the parsed declaration through a first compiler other than traditional compiler complying with the formal JAVA.TM language specification, wherein the intermediate language code block containing the instance of the generic classes is executable by a runtime engine (see for example, p.3, [0051] "some type of intermediate code"; also see Fig.7, step 3120 "Executable code representing the class method is executed" and related text).

But does not explicitly disclose the declared instance of the generic class is provided using a programming language other than JAVA language and generated based on the portion of JAVA language and the instance of the generic class developed in the programming language other than JAVA language. However, Getov in the same analogous art of programming in Java, discloses Java code can use high performance classes (libraries) written in other language (see for example, p.863, SUMMARY, "in this paper we discuss how new Java programs can capitalize on high-performance libraries for other language" and related descriptions). Getov also discloses a method of binding of native library to Java suing dynamically linking the library to the Java virtual machine or linking the library to the object code produced by a stand-alone Java compiler (see for example, p.864, section 2. "Creating Native Library Wrappers" and related text) .Therefore, it would have been obvious to one having ordinary

skill in the art at the time the invention was made to use/reference the other classes/libraries in a programming language (Fortran, C) other than Java language and generate executable code based libraries and Java language. One would have been motivated to do so to enable high-performance computation in Java by reference existing native libraries in other programming language.

Claim 14:

Hostetter discloses a method as recited in claim 13 further comprising associating the declaration of the instance of the generic class with a defined generic class in a generic class library (see for example, p.5, [0069], "The ClassTemplate provides source code representations"; also see p.5, [0071], "When the compiler encounters in the program code a variable declaration in which the declared type is a template-generated class").

Claim 15:

Hostetter also discloses a method as recited in claim 14 further comprising tokenizing a parse tree with an identifier corresponding to the defined generic class, the parse tree comprising a hierarchical representation of the declaration (see for example, Fig.4B, element 222 "identifier"; also see p.5, [0077]).

Claim 16:

Hostetter discloses a method as recited in claim 13 further comprising creating metadata describing the portion of the JAVA.TM. language software (see for example, p.4, [0055], "A method binding is an object that stores information about a class method").

Claim 17:

Hostetter discloses a method as recited in claim 14 further comprising validating an operation on the instance of the generic class based on the defined generic class (see for example, Fig.5, step 2085, "Compiler Error"; also see, p.6, [0090], "the compiler notifies the software programmer or end user of the compilation error…").

Claim 36:

Hostetter discloses a method of generating microprocessor-executable code comprising:

- receiving a portion of source code written in a language for which generic classes are unspecified, the portion of source code including a generic class declaration declaring a generic class, the generic class declaration including at least one associated class reference defining a constructed class of the generic class (see for example, p.2, [0016], "add class templates to the Java programming language"; p.2, [0018], "creating a representation of the class template"); and

- generating a module having microprocessor-executable instructions corresponding to the constructed class, (see for example, p.4, [0054], "executable code"), the module further having metadata describing the constructed class (see for example, p.4, [0055], "A method binding is an object that stores information about a class method").

But does not explicitly disclose the declared instance of the generic class is provided using a programming language other than JAVA language and generated based on the portion of JAVA language and the instance of the generic class developed in the programming language other than JAVA language. However, Getov in the same analogous art of programming in Java, discloses Java code can use high performance classes (libraries) written in other language (see for example, p.863, SUMMARY, "in this paper we discuss how new Java programs can capitalize on high-performance libraries for other language" and related descriptions). Getov also discloses a method of binding of native library to Java suing dynamically linking the library to the Java virtual machine or linking the library to the object code produced by a stand-alone Java compiler (see for example, p.864, section 2. "Creating Native Library Wrappers" and related text) .Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use/reference the other classes/libraries in a programming language (Fortran, C) other than Java language and generate executable code based libraries and Java language. One would have been motivated to do so to enable high-performance

computation in Java by reference existing native libraries in other programming language.

Claim 37:

Hostetter further discloses a method as recited in claim 36 wherein the microprocessor-executable instructions comprise intermediate language instructions (see for example, .p.3, [0051] "some type of intermediate code").

Claim 38:

Hostetter also discloses a method as recited in claim 36 wherein the microprocessor-executable instructions comprise Microsoft.TM. Intermediate Language instructions (see for example, .p.3, [0051] "some type of intermediate code").

Claim 39:

Hostetter also discloses a method as recited in claim 36 wherein the metadata comprises at least one of:

- method information indicating one or more methods implemented by the constructed class (see for example, p.4, [0055], "A method binding is an object that stores information about a class method");

Claim 40:

Incorporating with rejection to claim 1, Hostetter further discloses a method of compiling comprising:

- receiving a portion of source code written in a language for which generic classes are unspecified in a formal language specification, the portion of source code including a first class reference having at least one associated class reference referencing a class associated with the first class (see for example, Fig.5, step 2020 "Read Instruction From Program Code" and related text; also see p.2, [0016]-[0017], "Java programming language", "template generated class"); and
- generating an intermediate language representation of the portion of source code, the intermediate representation having an instance of the first class and an instance of the at least one associated class (see for example, Fig.5, step 2045, 2075, "Compile" and related text; also see p.5, [0071], "template-generated class representation, an object representation of the template-generated class in memory").

But does not explicitly disclose the first class being provided using a second programming language other than JAVA.TM language and generating language-neutral intermediate language code representing the portion of source code from JAVA. language and the first programming language. However, Getov in the same analogous art of programming in Java, discloses Java code can use high performance classes (libraries) written in other language (see for example, p.1, SUMMARY, "in this paper we discuss how new Java programs can capitalize on

high-performance libraries for other language" and related descriptions). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use/reference the other classes/libraries in a programming language (Fortran, C) other than Java language. It also would have been obvious in the computer art that said libraries/class can be saved remotely and called/referenced through a network. One would have been motivated to do so to enable high-performance computation in Java by reference existing native libraries in other programming language.

Claim 42:

Hostetter also discloses a method as recited in claim 40 wherein the language is a JAVA.TM. language (see for example, p.2, [0016], "java programming language").

Claim 43:

Hostetter also discloses a method as recited in claim 40 further comprising validating the type based on a definition of the first class (see for example, Fig.5, step 2085, "Compiler Error"; also see, p.6, [0090], "the compiler notifies the software programmer or end user of the compilation error...").

Claim 44:

Hostetter discloses a method as recited in claim 43 further comprising validating an operation on the first class based on a definition of the first class (see for example, Fig.5, step 2085, "Compiler Error"; also see, p.6, [0090], "the compiler notifies the software programmer or end user of the compilation error…").

Claim 45:

Hostetter also discloses a method as recited in claim 40 further comprising interpreting the intermediate representation for execution by a microprocessor (see for example, p.3, [0051] "some type of intermediate code"; also see p.4, [0054], "executable code").

Claim 50:

Hostetter further discloses a method as recited in claim 40 wherein the at least one associated class reference includes one or more nested generic class references (see for example, p.6, [0099] "other classes or class templates are inherited").

Claims 18-24:

Claims 18-24 are a computer product version of claimed method as discussed in claim 1-9 above, wherein all claimed limitations have been address and/or set forth in claims 1-9. It is well known in the computer art that claimed method can be stored and/or exercised on the computer-readable medium. Therefore, as the

references teach all the limitation of claims 1-9, they also teach the limitations of claims 18-24. Thus, they also would have been obvious.

Claims 25-30:

Claims 25-30 are a computer product version of claimed method as discussed in claims 40-50 above, wherein all claimed limitations have been address and/or set forth in claims 40-50. It is well known in the computer art that claimed methods could be stored and/or exercised on the computer-readable medium. Therefore, as the references teach all the limitation of claims 40-50, they also teach the limitations of claims 40-50. Thus, they also would have been obvious.

Claims 51-54:

Claims 51-54 are a system version of claimed methods as discussed in claims 40-5- above, wherein all claimed limitations have been address and/or set forth in claims 40-50. Therefore, as the references teach all the limitation of claims 40-50, they also teach the limitations of claims 40-50. Thus, they also would have been obvious.

12. Claims 11-12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hostetter (Hostetter et al., US 2005/0060695 A1) in view of Getov (Getov et al., "High-performance parallel programming in Java exploiting native libraries") and

in further view of Lurie (Johnthan Lurie, Product Snapshot: J#, J# provides Java develops a key for entering the .Net platform) and Allen (Eric Allen, "Diagnosing Java code: Java generics without the pain, part 1-part2)

Claims 11-12:

Hostetter, Allen and Getov disclose a method as recited in claim 1 wherein the framework, but does not explicitly discloses the framework is a .NET.TM. Framework. However, Lurie in the same analogous art of Java programming discloses Visual J#.Net Framework (see for example, p.1, first paragraph, "Visual J#.Net is Microsoft's Java development tool for the .Net Framework") Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use Visual J# to implement/perform above method is claim 10. One would have been motivated to do so to take advantage of rich APIs offered by the .Net Framework to perform similar tasks as suggested by Lurie (see for example, p.1, third paragraph)

13. Claims 31-35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hostetter (Hostetter et al., US 2005/0060695 A1) in view of Getov (Getov et al., "High-performance parallel programming in Java exploiting native libraries") and in view of APA (Admitted Prior Art, BACKGROUND section of the specification, [0003]-[0005]) and Allen (Eric Allen, "Diagnosing Java code: Java generics without the pain, part 1-part2)

Claim 31:

Incorporating with rejection to claim 1, Hostetter discloses a method for compiling in a framework the method comprising:

- Parsing the portion of source code into a parse tree comprising each instance of the one or more generic types in the portion of source code, wherein each instance of the one or more generic types comprises: the first type; and at least one instance of one of the plurality of second types associated with the first type (see for example, p.2, [0024] "requires compilation"); and
- generating an intermediate representation of the parse tree representing the parse tree. (see for example, p.3, [0051] "some type of intermediate code").

But does not explicitly disclose the generic class is .NET generic type are not specified in a formal definition of the first programming language, wherein the specified generic types are provided using a second programming language other than the first programming language. However, APA discloses that formal specifications for some languages, such as Java language, do not specify generic classes and .Net may provide generic class in framework (see for example, p.1, [0005]). Getov in the same analogous art of programming in Java, also discloses Java code can use high performance classes (libraries) written in other language (see for example, p.1, SUMMARY, "in this paper we discuss how new Java programs can capitalize on high-performance libraries for other language" and related descriptions). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use and

compile .Net generic class in Hostetter's teachings. One would have been motivated to do so to take full advantage of the .Net Framework to perform similar tasks as suggested by APA (see for example, p.1, [0005]) and to enable high-performance computation in Java by reference existing native libraries in other programming language.

Claim 32:

Hostetter further discloses the method of claim 31 also comprising importing metadata describing the first class and the at least one instance of the associated class (see for example, p.4, [0055], "A method binding is an object that stores information about a class method").

Claim 33:

Hostetter also discloses the method as recited in claim 31 further comprising tokenizing the parse tree with a token corresponding to the generic type (see for example, p.2, [0024] "The process for generating method bindings…").

Claim 34:

Hostetter also discloses the method as recited in claim 33 further comprising tokenizing the parse tree with at least one token corresponding to the at least on instance of one of the plurality of second types associated with the first type. (see for example, p.2, [0024] "The process for generating method bindings…")

Claim 35:

APA further discloses the method as recited in claim 31 wherein each of the one or more generic types is a NET.TM. generic class (see for example, APA , p.1, [0005]).

14. Claims 46-49 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hostetter (Hostetter et al., US 2005/0060695 A1) in view of Getov (Getov et al., "High-performance parallel programming in Java exploiting native libraries") and in view of Allen (Eric Allen Diagnosing Java code: Java generics without the pain, part1, part2)

Claim 46:

Hostetter discloses a method as recited in claim 40, but does not explicitly disclose angular brackets surround the at least one associated class reference. However, Allen in the same analogous art of Java generics discloses the angular brackets surround the at least one associated class reference (see for example, p.3, example code with "<>"). It is well known in the computer art that angular bracket is used to define template or generics class.

Claim 47:

Hostetter discloses a method as recited in claim 40 and also disclose a linked list (see for example, p.5, [0075], bindings are stored in the ClassObject as elements of a linked list), but does not explicitly disclose the first class is a Queue class. However, it is well known in the computer art that Queue class is included in the .Net Class library to represents a first-in, first-out collection of objects. Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use queue class to implement. One would have been motivated to do so to implement the bindings using framework provided existing queue class.

Claims 48-49:

Hostetter discloses a method as recited in claim 47, but does not explicitly disclose at least one associated class comprises at least one of:

- an int type; a string type; and a Queue type.

However, Allen in the same analogous art of Java generics discloses generic types (see for example, p.2-3, section "Generic types to the rescue", Listing4. "Referencing type parameters like ordinary types and example code <string>, <integer>). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use generic types to define generic class. One would have been motivated to do so to eliminate casts as suggested by Allen (see for example, p.2, section "Generic types to the rescue", a natural way to eliminate casts like the one above is to argument the Java type system").

Conclusion

15. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

16. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Zheng Wei whose telephone number is (571) 270-1059 and Fax number is (571) 270-2059. The examiner can normally be reached on Monday-Thursday 8:00-15:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature of relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571- 272-1000.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Z. W./
Examiner, Art Unit 2192

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192